20 VIDEOLEZIONI DI BASIC
PER IMPARARE COL VIC 20

- Unità a disco
- Come funziona un drive
- Uso e manutenzione dei dischetti
- File sequenziali e random
- Comandi DOS
- Uso dei file
- Videogioco n° 14

GRUPPO
EDITORIALE
JACKSON

14

COMMODORE VIC 20

VIDEO BASIC VIC 20
Pubblicazione quattordicinale
edita dal Gruppo Editoriale Jackson
Direttore Responsabile:
Giampietro Zanga
Direttore e Coordinatore
Editoriale: Roberto Pancaldi
Autore: Softidea - Via Indipendenza 88 - Como
Redazione software:
Francesco Franceschini, Enrico Braglia,
Fabio Calanca
Segretaria di Redazione:
Marta Menegardo
Progetto grafico:
Studio Nuovaidea - Via Longhi 16 - Milano
Impaginazione:
Silvana Corbelli
Illustrazioni:
Cinzia Ferrari, Silvano Scolari
Fotografie:
Marcello Longhini
Distribuzione: SODIP
Via Zuretti, 12 - Milano
Fotocomposizione: Lineacomp S.r.l.
Via Rosellini, 12 - Milano
Stampa: Grafika '78
Via Trieste, 20 - Pioltello (MI)
Direzione e Redazione:
Via Rosellini, 12 - 20124 Milano
Tel. 02/6880951/5

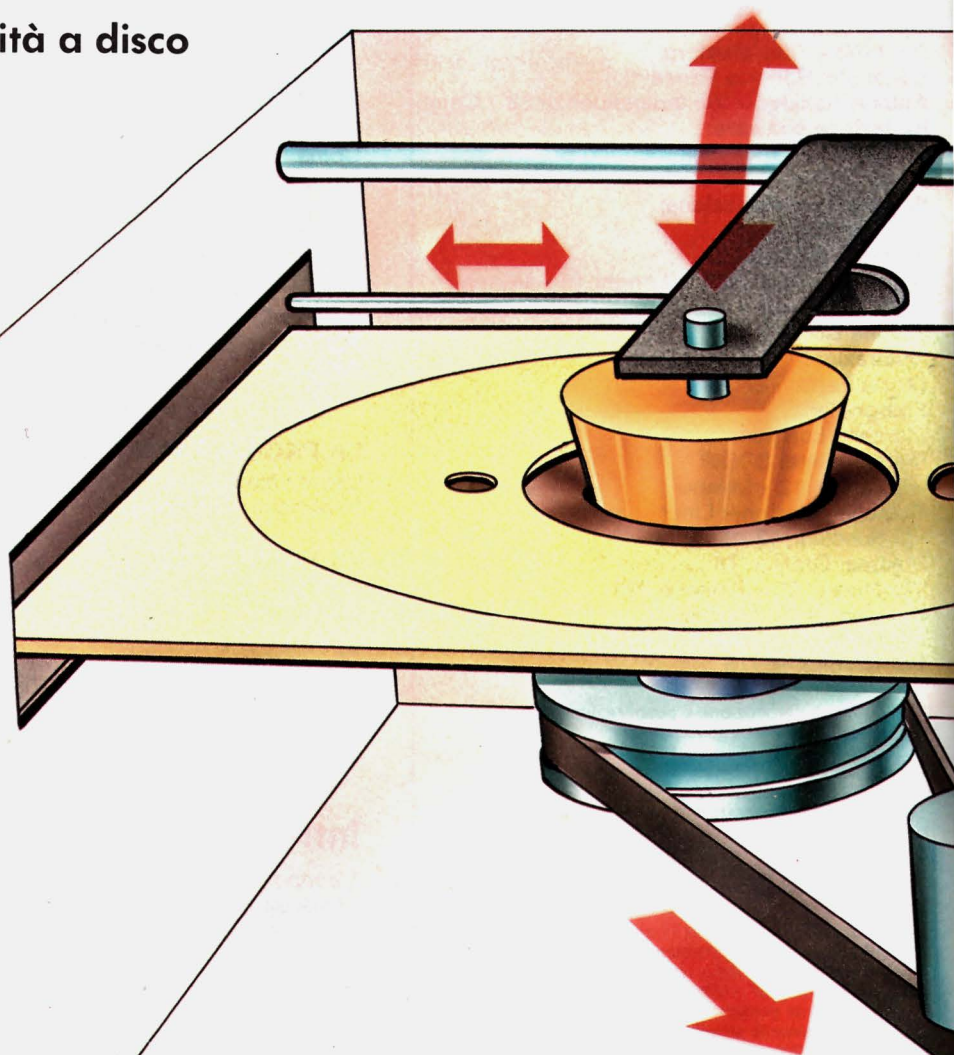
Autorizzazione alla pubblicazione Tribunale di Milano n° 422 del 22-9-1984
Spedizione in abbonamento postale Gruppo II/70 (autorizzazione della Direzione Provinciale delle PPTT di Milano).
Prezzo del fascicolo L. 8.000
Abbonamento comprensivo di 5 raccoglitori L. 165.000
I versamenti vanno indirizzati a: Gruppo Editoriale Jackson S.r.l. - Via Rosellini, 12 20124 Milano, mediante emissione di assegno bancario o cartolina vaglia oppure utilizzando il c.c.p. n° 11666203.
I numeri arretrati possono essere richiesti direttamente all'editore inviando L. 10.000 cdu. mediante assegno bancario o vaglia postale o francobolli.
Non vengono effettuate spedizioni contrassegno.

Gruppo Editoriale Jackson

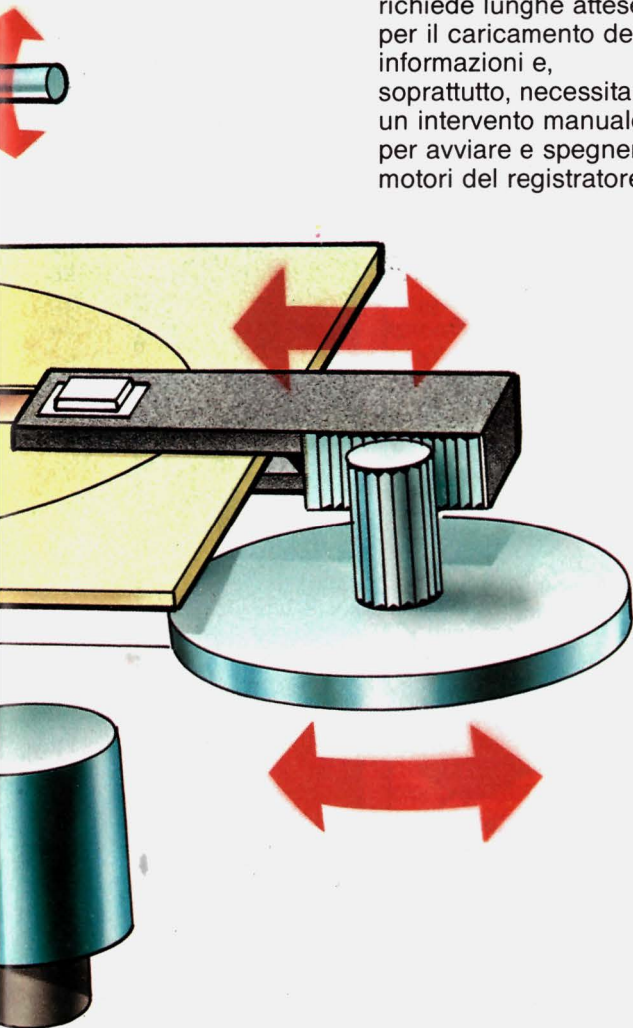
SOMMARIO

Introduzione

L'economico registratore è la memoria di massa per eccellenza addirittura si può dire che l'home computer non esisterebbe senza registratore.
Che noia però i lunghi e interminabili minuti per il caricamento o salvataggio di un programma, che estenuanti attese nella ricerca di un programma e che scarsa affidabilità, poi, in tutte le operazioni che vedono il registratore coinvolto.
Come sempre la tecnologia ha dato una risposta a tutti questi problemi: i floppy disk drive.
Trattare allora grosse moli di dati, in breve tempo e con una quasi totale affidabilità diventa semplice, il trucco è conoscere i comandi.

Unità a disco

Abbiamo visto nelle scorse lezioni come sia possibile - attraverso il registratore a cassette - memorizzare su nastro magnetico tutti i programmi e i dati che andrebbero altrimenti irrimediabilmente perduti spegnendo il calcolatore. La registrazione e la lettura da nastro - sicuramente avrai già avuto modo di accorgertene - è un'operazione molto

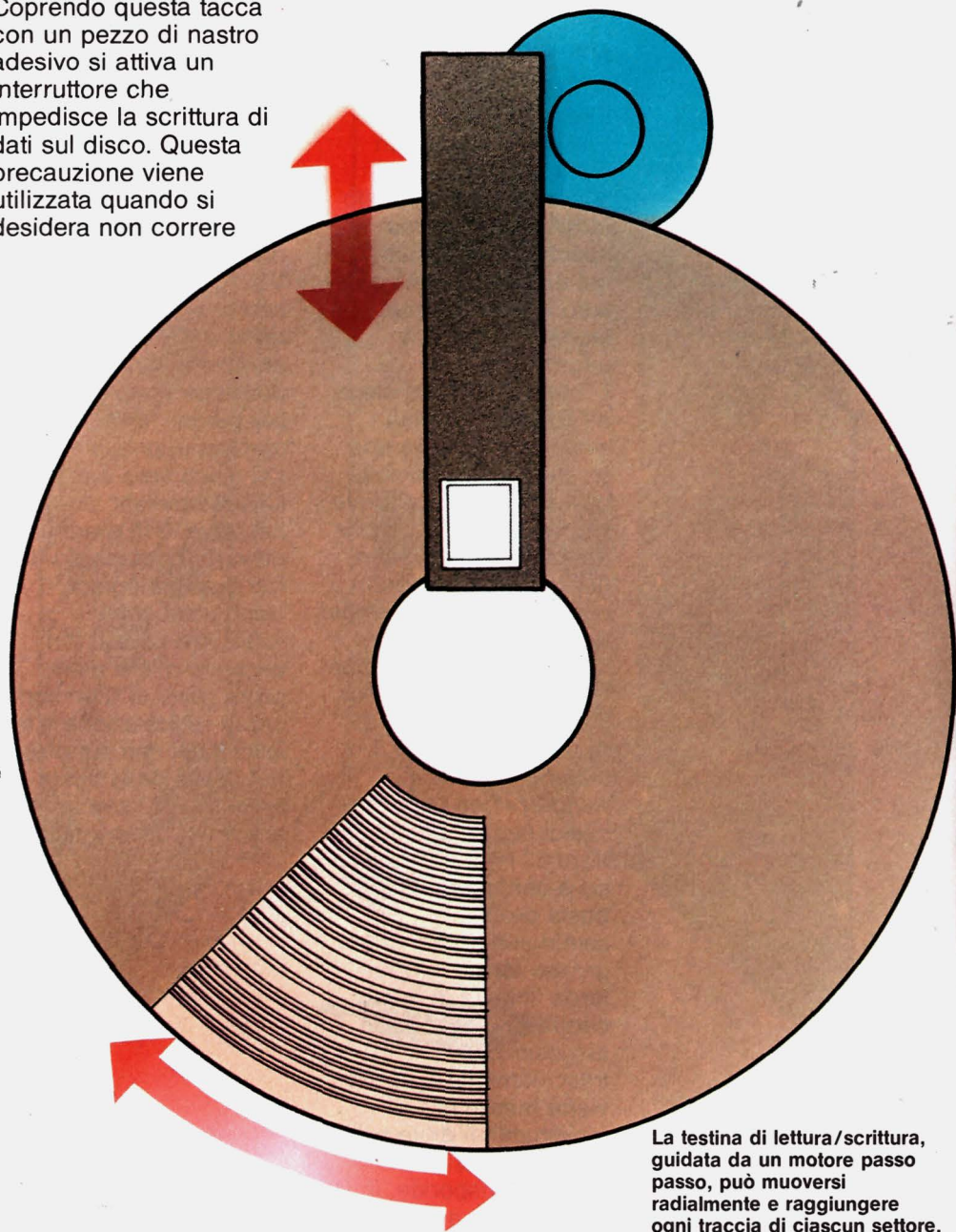
semplice, affidabile ed economica: tuttavia richiede lunghe attese per il caricamento delle informazioni e, soprattutto, necessita di un intervento manuale per avviare e spegnere i motori del registratore (il famoso "PRESS PLAY ON TAPE").

La ricerca dei programmi sul nastro è inoltre affidata a tecniche abbastanza empiriche, come il conteggio dei secondi o il rilievo del numero di giri compiuti dalla rotella di trascinamento, con il continuo rischio (soprattutto quando si è abbastanza distratti) di sovrapporre pezzi già registrati e provocare la perdita di programmi che magari avevano richiesto lunghe ore di paziente battitura.
Una delle più interessanti possibilità per l'espansione delle prestazioni offerte dal tuo VIC 20 è allora sicuramente costituita dall'unità a dischi flessibili (floppy disk).
Lo scopo dell'unità a disco (o "disk drive"), della quale ci occuperemo nella nostra lezione, è infatti proprio quello di eliminare tutti questi "inconvenienti", permettendo una registrazione altrettanto sicura di quella offerta dal registratore a cassette, ma molto più veloce e, soprattutto, completamente automatica.
Al contrario di quando si usa il registratore, infatti,

HARDWARE

non è necessario preoccuparsi in alcun modo del funzionamento del drive: la gestione delle varie operazioni - come per esempio la registrazione o il caricamento dei programmi - avviene in questo caso sotto il diretto controllo dell'unità centrale, senza più alcun intervento esterno di ricerca o di verifica.

Come funziona un drive

Il principio di funzionamento di un drive è basato su un disco magnetico, posto in rotazione da una specie di "giradischi" e sul quale una testina di lettura/scrittura può leggere o scrivere informazioni.
L'introduzione del disco all'interno dell'unità avviene attraverso una fessura, praticata nella parte anteriore del drive, immediatamente visibile date le sue dimensioni (leggermente superiori a quelle del disco: 5 pollici 1/4, circa 13 centimetri).
Il disco, chiamato anche "floppy disk" ("disco flessibile"), è costituito da un sottile foglio di materiale plastico, ricoperto con gli stessi ossidi metallici usati nei normali nastri magnetici, ed è contenuto in una busta protettiva semirigida dotata di un grosso foro e di una larga finestra. Il primo - centrale - serve per il dispositivo di trascinamento (quando viene messo in movimento il drive pone infatti in rotazione il disco; corrisponde, in un certo senso, al foro centrale dei dischi fonografici); la seconda, radiale, mette invece allo scoperto parte della superficie del disco, affinché la testina possa entrare in contatto con il supporto magnetico durante la rotazione del disco stesso.
Vicino al grande foro centrale si trova inoltre un forellino circolare attraverso il quale è possibile vedere uno o più piccoli fori corrispondenti, praticati nel disco vero e proprio. Una fotocellula, posta nel drive, è in grado di rilevare il passaggio dei fori quando il disco viene messo in rotazione. Questi fori vengono allora usati come punti di riferimento per la registrazione e la lettura dei dati sul disco.
Sul bordo della busta protettiva si trova inoltre una tacca rettangolare.

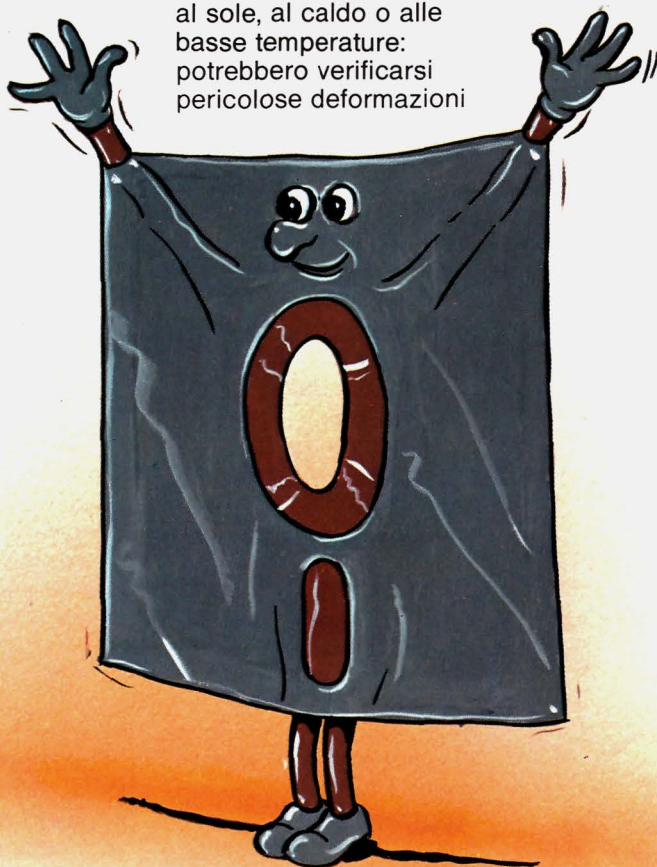
Coprendo questa tacca con un pezzo di nastro adesivo si attiva un interruttore che impedisce la scrittura di dati sul disco. Questa precauzione viene utilizzata quando si desidera non correre

La testina di lettura/scrittura, guidata da un motore passo passo, può muoversi radialmente e raggiungere ogni traccia di ciascun settore.

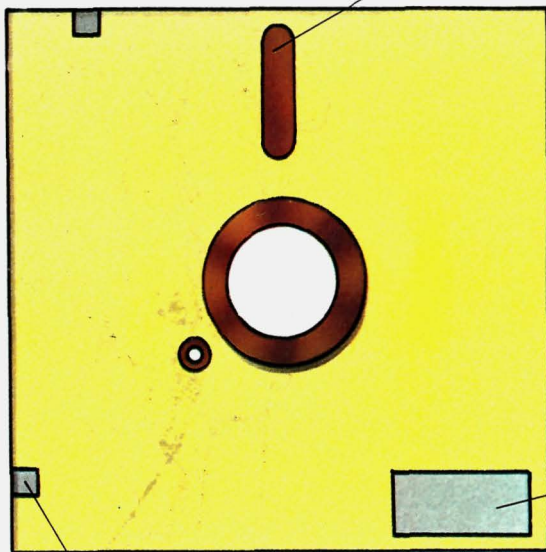
rischi di cancellazioni o alterazioni accidentali dei dati sul disco.
All'interno del drive sono presenti i motori per la rotazione del disco e per lo spostamento della testina, oltre naturalmente all'insieme dei circuiti necessari per pilotarli ed all'interfaccia per comunicare - attraverso il cavetto di collegamento - con l'unità centrale.
Il sistema di controllo dei dischi è inoltre costituito in modo da definire su ciascun dischetto vergine (cioè non ancora registrato) una struttura complessa, ma esattamente definita. Il floppy disk, infatti, quando è nuovo non è immediatamente utilizzabile: su di esso devono essere preventivamente registrate alcune informazioni indispensabili per il suo successivo funzionamento.
Affinché possa leggere un disco il drive richiede quindi che lo stesso abbia subito la cosiddetta "formattazione", cioè un'operazione che registri sul disco stesso una sorta di "mappa geografica" che la testina di lettura/scrittura utilizzerà come sistema di riferimento per orientarsi nella registrazione o nella lettura delle informazioni.
Senza addentrarci troppo in inutili dettagli tecnici, è sufficiente sapere che durante la formattazione il disco viene suddiviso in una serie di invisibili tracce concentriche, le quali sono a loro volta ulteriormente scomposte nei cosiddetti settori, che prendono questo nome proprio perché sono rappresentabili esattamente come settori delle circonferenze corrispondenti alle tracce.
Complessivamente, alla fine della formattazione, il disco risulta costituito questi 683 settori solo dei quali in grado di contenere 256 byte. Di questi 683 settori, solo 664 sono tuttavia effettivamente utilizzabili (gli altri 19 servono infatti al drive per memorizzarvi alcune informazioni a suo esclusivo uso e consumo): la reale capacità di memorizzazione di un dischetto è quindi pari a circa 170000 byte (664 * 256).

Manutenzione dei dischetti

Abbiamo visto che, nonostante le ridotte dimensioni, un floppy disk è in grado di registrare con precisione una grandissima quantità di informazioni; per questa ragione è quindi di importanza capitale osservare scrupolosamente alcune semplici, ma indispensabili precauzioni, analoghe a quelle già viste a proposito dei nastri magnetici, ma ancora più importanti vista la relativa delicatezza dei

dischetti:

- non toccare mai la superficie magnetica dei dischetti (attraverso il foro ovale). Sulle dita è sempre presente un sottile strato di grasso, che può impedire il corretto contatto tra testina e dischetto;
- evitare che la polvere si depositi sui dischetti: anche un microscopico granello di polvere o di cenere di sigaretta può infatti rovinare irrimediabilmente lo strato magnetico. La soluzione migliore è quella di riporre il dischetto nella propria busta di protezione non appena terminato di usarlo;
- non esporre i dischetti al sole, al caldo o alle basse temperature: potrebbero verificarsi pericolose deformazioni del supporto plastico;
- per quanto il termine "disco flessibile" possa aprire la strada alle più disparate interpretazioni, evitare nel modo più assoluto di sollecitare meccanicamente i dischi, curando di non maneggiarli con eccessiva sicurezza, di

non sovrapporli e di non schiacciarli con pesi di alcun genere;
- allontanare il più possibile i dischi da calamite, televisori, altoparlanti e in generale da qualunque apparecchiatura elettrica;
- tenere una copia di riserva di tutti i dischetti contenenti informazioni importanti, conservandola in un luogo diverso da quello in cui si trovano gli originali;
- quando il computer è spento non lasciare mai il dischetto nel drive: estrarlo sempre prima di spegnere la macchina e riporlo immediatamente.

Prima di concludere il nostro discorso è bene ricordare che conviene sempre acquistare dischetti di buona marca a doppia densità di registrazione (doppia densità significa che la qualità del supporto magnetico è in grado di rispondere nel migliore dei modi alle più piccole sollecitazioni impartite dalla testina del drive, con la quasi assoluta sicurezza e garanzia di non perdere -a causa di eventuali difetti dovuti alla grossezza della grana- alcun dato o informazione).

Il DOS

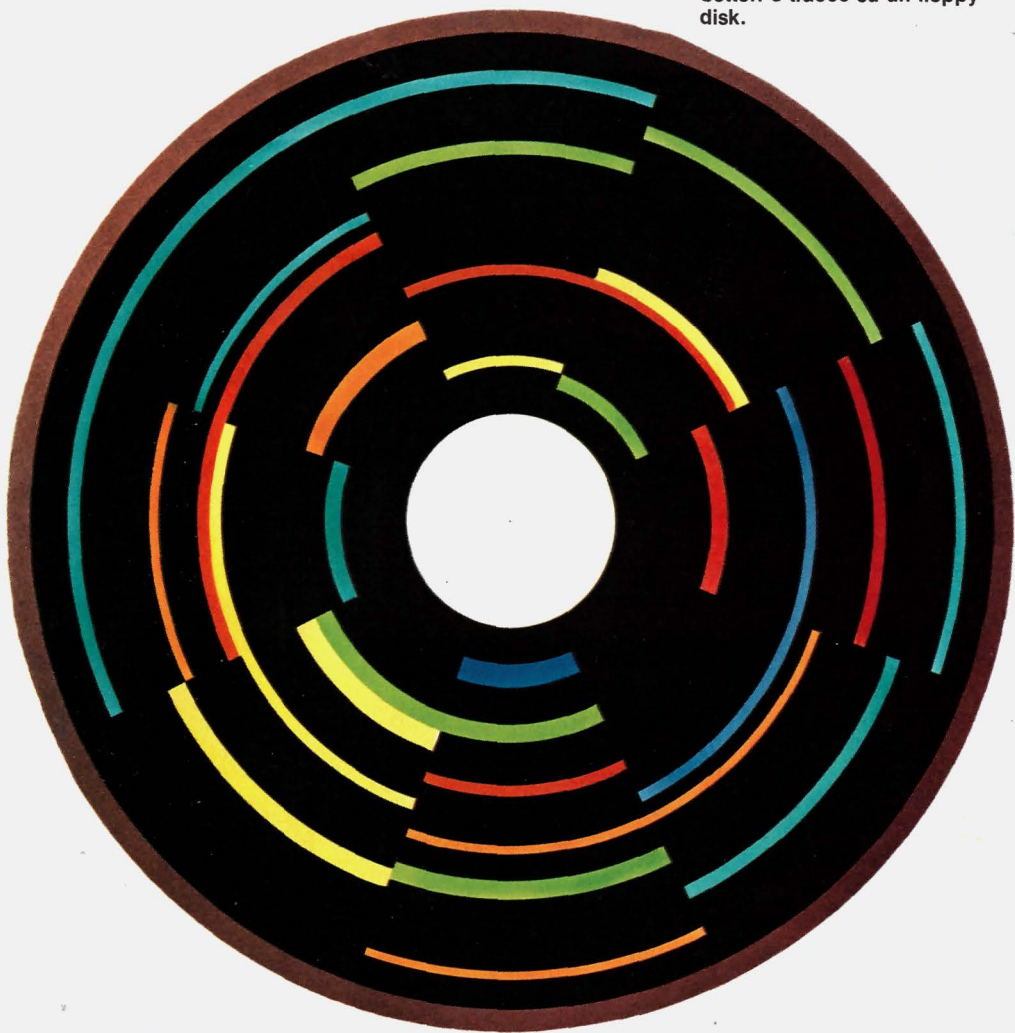
Adesso che ti sei fatto un'idea del funzionamento dell'unità a dischi, è necessario fermarci un momento per parlare di come possa avvenire lo scambio delle informazioni tra l'elaboratore e il disk-drive.
La gestione dell'unità a dischi è affidata a un particolare programma, chiamato DOS (abbreviazione di Disk Operating System, sistema operativo del disco), che si occupa di coordinare le molteplici attività ed operazioni eseguibili dal drive.
La funzione del DOS è sotto certi aspetti molto simile a quella dell'interprete BASIC, dato che permette di impartire e di controllare

- attraverso pochi e brevi comandi di facile comprensione - operazioni tutt'altro che semplici, facendosi quindi carico delle parti più noiose e ripetitive.
In effetti, l'intero funzionamento di un elaboratore è sempre completamente gestito da un sistema operativo, che controlla e coordina in continuazione (senza che l'utilizzatore abbia la minima possibilità di accorgersene) il funzionamento di tutto il sistema.
Nel caso specifico il DOS è la parte di sistema operativo che si occupa dell'unità a dischi.
Per quanto riguarda il VIC 20 il DOS risiede in una zona di memoria ROM posta all'interno del disk-drive; la parte di sistema operativo, residente nella memoria centrale, che deve gestire i dischi è quindi ridotta alla sola interfaccia, di tipo seriale, con l'unità a dischi: tutto quanto serve alla effettiva gestione del drive è eseguito all'interno dell'unità stessa.
All'unità centrale resta solo il compito di comunicare i comandi all'unità periferica e leggere le risposte.
Quando si dà un comando per il drive viene trasmesso sul cavo di collegamento seriale; il drive lo riconosce ed inizia ad eseguirlo. Nel caso in cui il comando non implichi ulteriori scambi tra unità centrale e periferica l'unità centrale, una volta accertata la ricezione del comando, continua l'esecuzione del programma in corso; in caso contrario attende le informazioni in risposta dal drive. Se si tenta di inviare un altro comando al disco, questo viene eseguito solo al termine del comando precedente, poiché la ricezione stessa del comando risulta bloccata fintanto che l'unità periferica è impegnata nell'esecuzione di un altro comando.
Questo schema di procedimento, che permette di migliorare le prestazioni di tutto l'insieme, è completamente controllato dal DOS. Più avanti impareremo i comandi riconosciuti dal DOS.

I file

I programmi sono abbastanza utili, ma, diciamo la verità, quello che realmente si vuole dal calcolatore è trattare informazioni, dati, numeri, nomi, indirizzi, quantità.
Qualunque istruzione che possa essere scritta può anche essere inserita nella memoria del computer e, di conseguenza, memorizzata sul drive.
Un file è un insieme di informazioni che, per un motivo qualunque, si desidera raggruppare.
Esempi di file possono essere: i nomi degli alunni di una classe, i numeri estratti durante una partita di tombola, i nomi e gli indirizzi dei nati in un certo giorno, un programma. Il DOS mette a disposizione principalmente due tipi di file:
— file sequenziali: sono file in cui le informazioni vengono registrate una di seguito all'altra, in modo sequenziale. Sono i più semplici da utilizzare, ma non permettono una grande flessibilità di uso;
— file random (ad accesso diretto): in questo caso le informazioni sono scritte (o lette) in un punto qualsiasi del file.
Rispetto al file sequenziale il file random è più pratico ed efficiente, ma ha una struttura leggermente più rigida e quindi più complicata da gestire.
Esamineremo quindi come dovremo fare per memorizzare su disco - raccogliendoli tra loro - tutti i dati e le informazioni che desideriamo non siano irrimediabilmente perduti spegnendo il calcolatore.
Per fare questo è comunque inizialmente necessario introdurre due nuovi termini, usatissimi quando si parla di file: Record e Campi.
Facciamolo (per essere più chiari possibile) attraverso un esempio.
L'elenco telefonico della tua città, preso nel suo insieme, è un file: esso contiene infatti informazioni ordinate e classificate secondo un ordine ben preciso.
Questo file è formato da migliaia di record, ciascuno dei quali costituito da un cognome, un nome, un indirizzo e un numero di telefono. Un record può essere quindi definito come un gruppo di dati correlati fra loro; le singole registrazioni che si susseguono nel

record si chiamano invece campi.
Nel nostro caso esistono 4 campi per ciascun record: un campo Cognome, un campo Nome, un campo Indirizzo e un campo Numero di telefono.
La scelta del formato del *record (cioè del numero* di campi) è naturalmente arbitraria: avremmo potuto considerare - oltre ai campi appena visti - *anche un campo* Numero di prefisso, indicante appunto il prefisso telefonico.
Abbiamo detto che nei file sequenziali (come dice il nome stesso) le informazioni sono

Settori e tracce su un floppy disk.

registrate una di seguito all'altra. Ciò significa che per leggere (o scrivere) l'ultima informazione del file occorre aver prima letto (o scritto) tutte le informazioni precedenti. Questa limitazione può sembrare un notevole svantaggio (sarebbe alquanto scomodo se anche noi per cercare "Rossi Mario" nell'elenco telefonico dovessimo cominciare dalla prima pagina), tuttavia - grazie alla notevole velocità di lettura dei dati sul drive - entro file di dimensioni abbastanza contenute il tempo di ricerca di un singolo record è abbastanza ragionevole. Verificheremo e approfondiremo comunque il nostro discorso nella parte della lezione dedicata alla programmazione.

Comandi DOS

Quando il tuo VIC 20 è collegato a una o piùù unità a dischi molti comandi che erano già disponibili per il controllo del registratore diventano validi anche per il disk drive. Altre istruzioni, invece, sono caratteristiche della sola unità a disco. Tutti i comandi che consentono di operare con i dischi possono comunque essere utilizzati, come accadeva per il registratore, sia in *modo immediato* che da programma.

SAVE

Il significato dell'istruzione SAVE ti è già ben noto da parecchio tempo, poichè ti consente di trasferire i programmi dalla memoria del computer al nastro magnetico del registratore.
Per poter essere usato con l'unità a disco questo comando richiede tuttavia di essere impartito con un pochino di attenzione in più, dato che occorre specificare, oltre al nome del programma, anche l'unità periferica verso la

quale il programma deve essere inviato.
Vediamo subito un esempio:

```
SAVE "PIPPO", 8
```

memorizza il programma attualmente in memoria sull'unità a disco, battezzandolo con il nome PIPPO, che PIPPO, come nel caso del registratore, è il nome sotto cui si

desidera registrare il programma: deve essere una stringa di lunghezza non superiore ai 16 caratteri.
Il numero 8 è invece il numero che contraddistingue il disk drive: esso viene predisposto in fabbrica dalla Commodore e non può essere modificato.
Tutti i comandi rivolti verso il disco devono contenere questo numero.
Se ciò non viene fatto, il computer presuppone che il programma da salvare debba essere inviato al registratore, dando così inizio all'intera procedura di memorizzazione su

nastro (comparsa sullo schermo di "PRESS PLAY ON TAPE", eccetera ...).
Il drive, prima di registrare il programma, controlla che sul disco non esista già un nome uguale a quello assegnato al nuovo file: se il nome esiste, il comando SAVE non viene eseguito e si avrà il lampeggio del LED rosso del drive per segnalare l'errore.
Ogni volta che il drive incorre in un errore di qualsiasi tipo l'unica segnalazione che appare è infatti quella del LED: occorre quindi stare molto attenti, visto che in apparenza il comando sembra essere stato eseguito, mentre in realtà nulla è accaduto.
Per poter scrivere un file (con il termine file, ti ricordo, si intende un qualsiasi insieme di informazioni, quindi anche un programma è un file) con lo stesso nome di uno già esistente sul disco, perdendo logicamente quello vecchio, si può allora dare questo comando

```
SAVE "@ 0 : PIPPO", 8
```

Infatti il carattere "@" fa in modo che il nuovo file vada a sostituire quello vecchio con lo stesso nome, prendendone il posto. È comunque consigliabile non usare troppe volte il comando SAVE in questo formato, dato che si può danneggiare la directory del disco, cioè l'archivio entro cui il drive scrive i nomi dei file presenti sul disco stesso. Vedremo più avanti che esiste una specifica istruzione in grado di cancellare i file che non servono più. Si potrà quindi cancellare innanzitutto il vecchio programma e quindi memorizzare il nuovo file.

Sintassi del comando

SAVE "nome del programma", 8

VERIFY

Dopo il comando SAVE è consigliabile usare sempre il comando VERIFY per controllare che il programma memorizzato su disco sia identico a quello presente in memoria. Supponendo di aver memorizzato il file PIPPO, l'istruzione va scritta così:

```
VERIFY "PIPPO", 8
```

permettendo di verificare se la memorizzazione del programma è stata effettuata correttamente.

Se qualcosa non sarà andato per il verso giusto, potrai sempre ripetere l'operazione di memorizzazione, evitando di perdere tutto quanto si trovava in memoria a causa di un banale errore o malfunzionamento.

Sintassi del comando

VERIFY "nome del programma", 8

LOAD

Adesso che abbiamo salvato il programma su disco vogliamo imparare anche come fare per poterlo caricare nuovamente nella memoria. L'istruzione adibita a questo scopo è ancora una volta identica a quella usata col registratore a cassette. Quindi, se volessimo recuperare il programma che avevamo scritto sul disco col nome PIPPO, dovremmo impartire:

```
LOAD "PIPPO", 8
```

Dopo pochi secondi il programma si troverebbe nella memoria, pronto per essere eseguito. L'esempio che abbiamo appena fatto è tuttavia molto particolare: eravamo infatti sicuri che sul disco si trovava il programma PIPPO e lo abbiamo quindi potuto caricare senza alcuna incertezza.
Altre volte, invece, l'elenco dei file presenti sul disco ci è ignoto: sarebbe quindi comodo poter disporre di una sorta di catalogo nel quale leggere i file che sono stati registrati sul disco.
Abbiamo detto che il disco può essere immaginato come suddiviso in anelli concentrici, detti tracce: nella traccia numero 18 il computer memorizza proprio l'indice dei programmi residenti sul disco, chiamato DIRECTORY.
Questo indice è un file come tutti gli altri e quindi può essere caricato con l'istruzione LOAD. Il comando da impartire è il seguente:

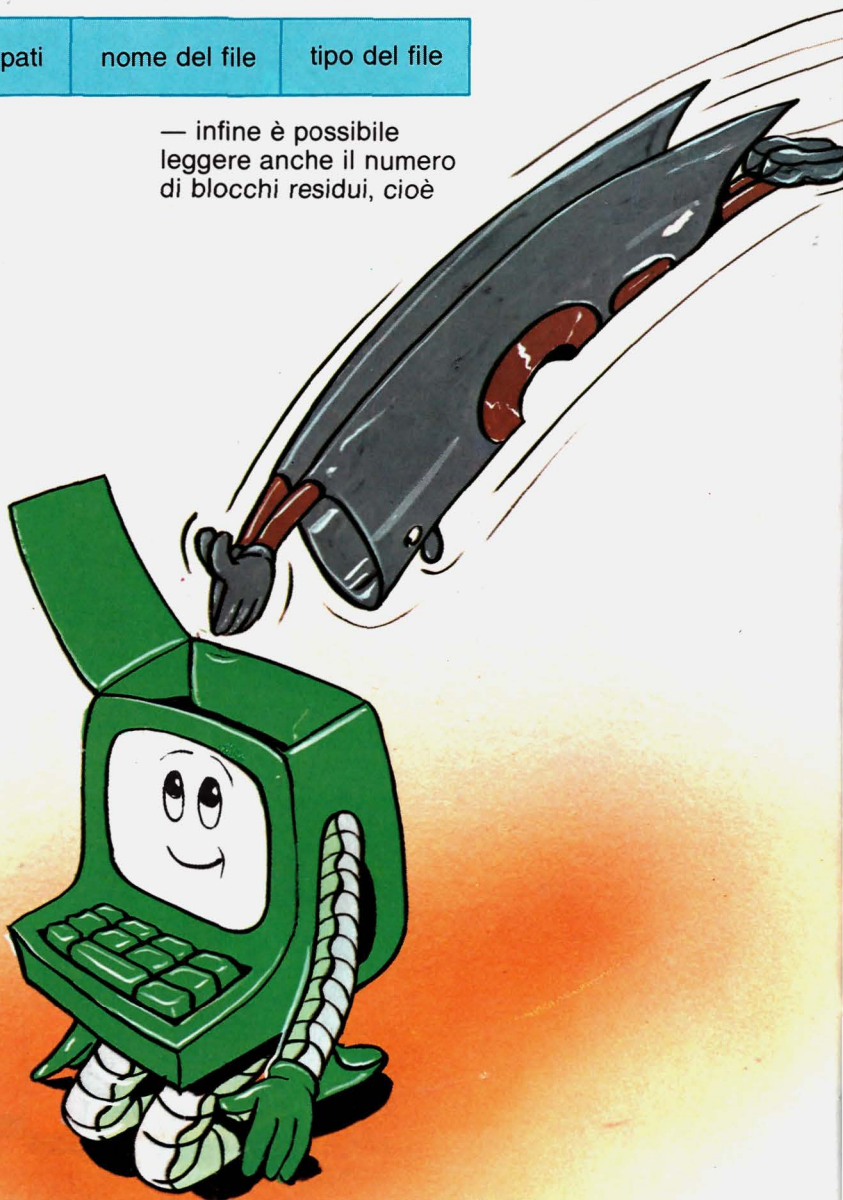
```
LOAD "$", 8
```

La luce rossa si accenderà per un istante e la directory verrà caricata in memoria. Una successiva istruzione LIST ti permetterà di vederne il contenuto:
— la prima riga contiene in reverse il nome assegnato al disco nella istruzione di formattazione (vedi più avanti);

LINGUAGGIO

— nelle righe seguenti sono assegnati tutti i file registrati su disco secondo l'ordine:

blocchi occupati	nome del file	tipo del file

— infine è possibile leggere anche il numero di *blocchi residui*, cioè non ancora memorizzati e perciò disponibili per la memorizzazione di altri dati.

Oltre a permetterti di caricare i programmi nel modo che abbiamo già visto, è possibile utilizzare LOAD senza specificare per intero il nome del programma, in quanto, grazie all'opzione chiamata "PATTERN MATCHING", la parte terminale del nome può essere sostituita con il carattere "*".
Così, per esempio,

```
LOAD "PRO*", 8
```

provocherà il caricamento in memoria del primo programma avente come iniziali PRO.
Se però sul disco sono presenti due programmi che iniziano così:

```
PROVA DI STAMPA
PROGRAMMA 1
```

e desideri caricare il secondo, l'istruzione corretta sarà:

```
LOAD "PROG*", 8
```

Il comando LOAD "*", 8 causerà invece il caricamento dell'ultimo programma scritto o caricato sul disco; se prima del comando non sarà stato caricato o memorizzato alcun programma, si otterrà allora il primo presente sull'elenco della directory.
È anche possibile sostituire alcune lettere del *nome con il carattere* "?". Per esempio:

```
LOAD "P??G*", 8
```

provocherà la sostituzione di ?? con RO e caricherà il secondo programma. Se tuttavia sul disco esistesse un altro programma con il nome:

```
PRIGIONE
```

il comando sarebbe ambiguo, visto che non specifica esattamente qual è il programma, tra PROGRAMMA 1 e PRIGIONE, che desideri caricare. In questo caso viene caricato il primo presente nell'indice della directory.

Sintassi del comando

LOAD "nome del programma", 8

NEW

Nei discorsi che finora sono stati fatti abbiamo sempre supposto di *avere già disponibile un* dischetto formattato. Adesso vedremo quindi le operazioni necessarie per poter preparare un dischetto vergine per le successive registrazioni. Innanzi tutto occorre accertarsi che il disco da formattare non contenga altri programmi (dal momento che questa operazione li distruggerà) e che non abbia la tacca di protezione dalla scrittura coperta con nastro adesivo.
A questo punto basta dare le seguenti istruzioni

```
OPEN 15, 8, 15
PRINT # 15, "NEW0 : NOME, 12"
```

oppure

```
PRINT # 15, "N0 : NOME, 12"
```

— la prima istruzione apre il canale (ti ricordi cosa sono i canali?) numero 15 verso la periferica numero 8 (il disk drive), con indirizzo secondario 15 (che il drive utilizza solo per trasmettere o ricevere informazioni);
— la seconda provoca la *formattazione vera e* propria del disco, ossia la sua suddivisione per tracce o settori. È un'operazione che richiede circa un minuto di tempo.
NEW serve per indicare che il dischetto deve essere trattato come se fosse nuovo e quindi formattato;
— il numero 0 sta ad indicare che la formattazione va fatta sul primo drive. In un sistema dotato di due drive tale numero, per avere la formattazione sul secondo drive, avrebbe dovuto essere 1;
— il nome è una stringa alfanumerica che puoi decidere a tuo piacere, *lunga al massimo 16* caratteri, mentre il 12 è un numero qualsiasi di due cifre (avremmo cioè potuto mettere 36 o 89). Entrambi sono utili al drive per la classificazione del dischetto.
Come al solito (avevamo già visto la cosa parlando della stampante), terminata l'operazione di formattazione occorre chiudere il canale che avevamo aperto:

```
CLOSE 15
```

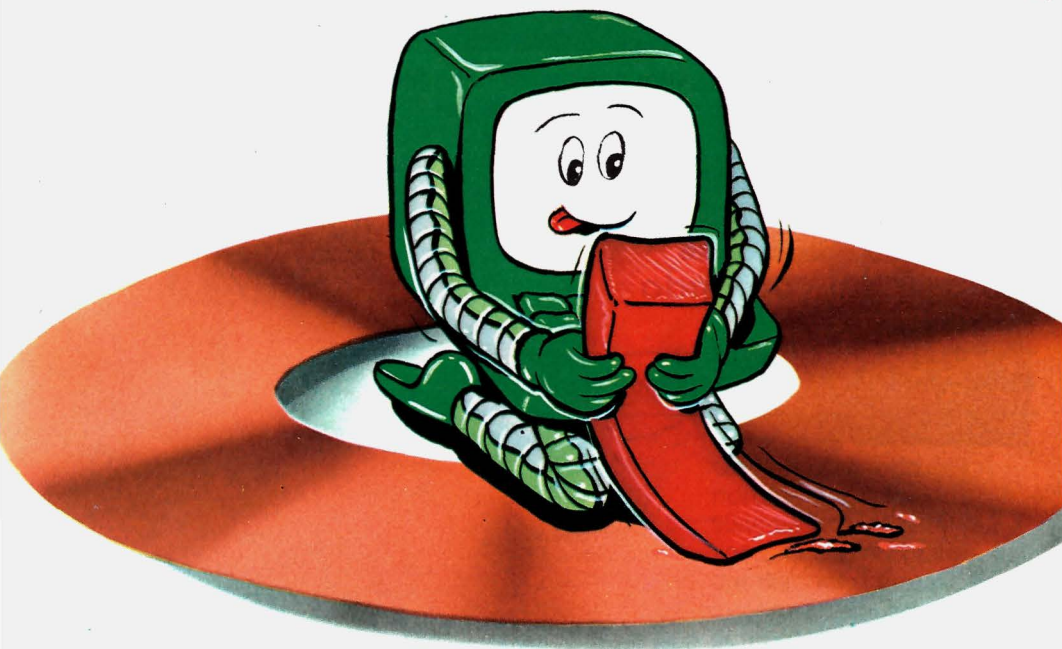
A questo punto il disco è pronto per registrare le informazioni che vorrai.

Sintassi del comando

OPEN 15, 8, 15

PRINT # 15, "NEW0 : nome, 12"

SCRATCH

Questo comando serve per cancellare file dal dischetto, rendendo nuovamente disponibile per altre operazioni lo spazio che essi occupavano in precedenza.
Anche in questo caso, prima del comando, è necessario aprire il canale verso l'unità a disco. Supponendo di voler cancellare il nostro programma PIPPO, il

comando dovrà quindi essere:

```
OPEN 15, 8, 15
PRINT # 15, "SCRATCHO : PIPPO"
```

oppure, in forma abbreviata

```
OPEN 15, 8, 15
PRINT # 15, "SO : PIPPO"
```

Al solito, finita l'operazione, si chiude il canale: CLOSE 15.
Anche in questo caso è possibile sfruttare la PATTERN MATCHING per eliminare più file contemporaneamente:

```
OPEN 15, 8, 15
PRINT # 15, "SO : PIP*"
```

cancella quindi tutti i file il cui nome inizia per PIP.

RENAME

Il comando RENAME consente di cambiare il nome a un file; in pratica, modifica soltanto il nome nella directory, senza toccare il punto in cui si trovava il programma vero e proprio. Per questa ragione è un'operazione che richiede pochissimo tempo per poter essere eseguita dal drive.
Per esempio, se sul disco avessimo un file

Sintassi del comando

OPEN 15. 8, 15

PRINT # 15, "SCRATCHO : nome del programma"

chiamato PROVA e volessimo ribattezzarlo DEFINITIVO, potremmo fare:

```
OPEN 15, 8, 15
PRINT # 15, "RENAME0 : DEFINITIVO = PROVA"
CLOSE 15
```

oppure, in forma abbreviata

```
OPEN 15, 8, 15
PRINT # 15, "R0 : DEFINITIVO = PROVA"
CLOSE 15
```

Sintassi del comando

OPEN 15, 8, 15

PRINT#15,"RENAME0: nuovo nome=vecchio nome"

VALIDATE

A differenza del nastro magnetico, il disco ha una organizzazione di dati molto complicata, con pezzi di programma disposti lungo le varie tracce e settori.
Talvolta può quindi accadere che - a forza di cancellazioni, registrazioni ed aggiornamenti - la somma dei settori liberi ed occupati rilevabile dalla lista della directory non dia 664, come invece dovrebbe accadere. Per mettere ordine in un disco dove si verifichino queste situazioni irregolari si utilizza quindi il comando VALIDATE:

```
OPEN 15, 8, 15
PRINT # 15, "VALIDATE"
CLOSE 15
```

oppure, in forma abbreviata

```
OPEN 15, 8, 15
PRINT # 15, "V"
CLOSE 15
```

Questo comando riordina il disco, sistemando le varie zone, libere ed occupate, del dischetto stesso. È buona norma impartire di tanto in tanto una VALIDATE: così facendo

si eviteranno possibili errori o problemi.

Sintassi del comando

OPEN 15, 8, 15

PRINT # 15, "VALIDATE"

INITIALIZE

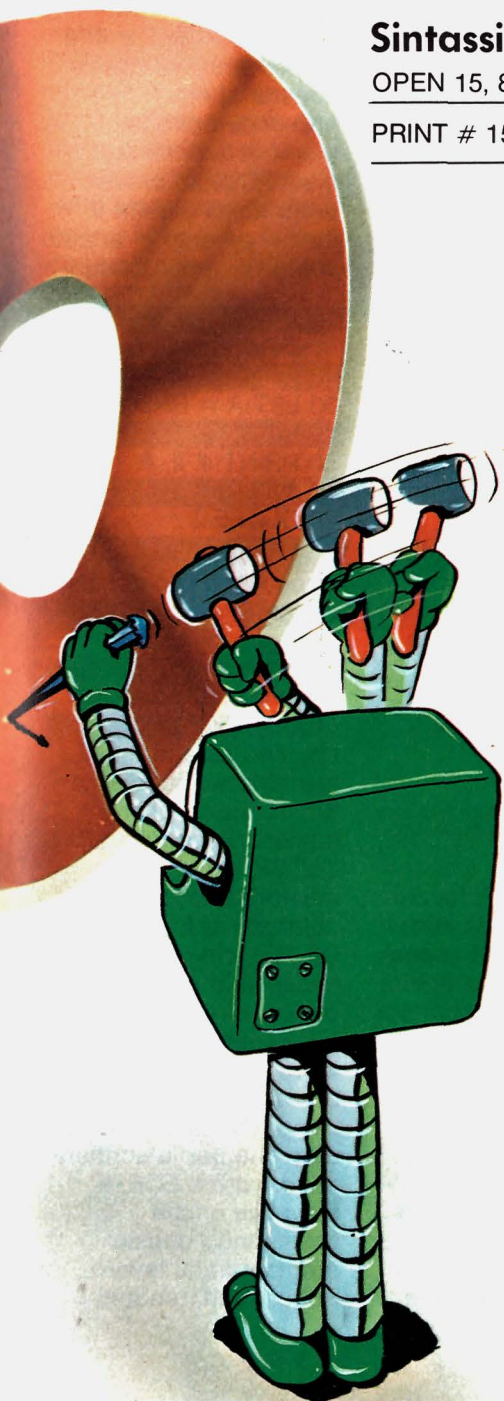
Questo comando serve, come dice il nome stesso, per inizializzare il dischetto nel drive. L'operazione di inizializzazione viene normalmente compiuta dal tuo drive al momento dell'accensione: se osservi l'unità, vedrai infatti la luce rossa accendersi per un brevissimo tempo e sentirai girare il dischetto.
Tuttavia, può accadere che dopo una condizione di errore dell'unità a disco alcune operazioni risultino impedite.
INITIALIZE riporta quindi il disk drive nelle stesse condizioni in cui si trova al momento dell'accensione.
Teoricamente, in alternativa a questo comando, è possibile spegnere o riaccendere il drive (l'operazione non è molto elegante, ma è altrettanto efficace):

```
OPEN 15, 8, 15
PRINT # 15, "INITIALIZE"
```

oppure, in forma abbreviata:

```
OPEN 15, 8, 15
PRINT # 15, "I"
CLOSE 15
```

Sintassi del comando

OPEN 15, 8, 15

PRINT # 15, "INITIALIZE"

Usare i file sequenziali

In questa parte della lezione approfondiremo il discorso che poco fa avevamo cominciato a proposito dei file. Innanzitutto, prima di poter accedere a un file bisogna aprirlo.
Aprire un file significa comandare al DOS di ricercare delle informazioni relative a questo file: se esso è già sul disco e, se lo è dove si trova esattamente.

Il comando OPEN fa eseguire al drive tutte queste operazioni, riservando anche una certa area nella memoria del computer che verrà usata come buffer (detta anche memoria tampone). Il buffer permette al drive di accedere solo saltuariamente al disco e non ogni volta che il programma ha un singolo dato da leggere o da scrivere. In altre parole, l'accesso al disco avviene per blocchi di informazioni e non per singolo dato, con conseguente grande risparmio di tempo. Un esempio di comando OPEN per l'apertura di un file sequenziale è il seguente:

```
OPEN 2, 8, 2, "PROVA, S, W"
```

I caratteri S e W dopo il nome del file indicano che questo dovrà essere di tipo S(equenziale) e di modo W, cioè di scrittura (dall'inglese W (rite)). In altre parole, il file viene aperto per essere scritto se avessimo voluto aprirlo per leggerne il contenuto avremmo dovuto mettere, anziché W, una R (R(eading) in inglese).
A questo punto tutte le istruzioni PRINT # che si riferiranno al canale appena aperto provocheranno la memorizzazione sul disco. Questo breve programma illustra il funzionamento di quanto abbiamo appena detto:

```
10 OPEN 2, 8, 2, "PROVA,
   S, W"
20 FOR I = 1 TO 200
30 LET A = RND (0)
40 PRINT A
50 PRINT # 2, A
60 NEXT I
70 CLOSE 2
```

Dopo l'apertura del file verranno inviati 200 numeri casuali rispettivamente al video e al drive; alla fine del ciclo il file sarà chiuso e i numeri saranno stati memorizzati sul disco. Subito dopo l'ultimo numero, in seguito alla CLOSE, verrà inoltre memorizzato un carattere di EOF (End OF File, cioè fine del file) necessario al computer per riuscire a gestire in futuro quel file, quando vorrai recuperare ciò che vi era scritto.
L'importanza di CLOSE, oltre che per la scrittura del carattere EOF, è rilevabile anche osservando questo programma al lavoro. Potrai infatti renderti

conto che la spia rossa del drive non rimarrà accesa in continuazione: le informazioni verranno infatti registrate solo quando il buffer sarà stato man mano riempito nel corso dell'esecuzione. Se non avessimo inviato la CLOSE, i dati rimasti, contenuti nel buffer al termine del ciclo, non avrebbero mai raggiunto il dischetto, visto che il computer si sarebbe aspettato nuovi "arrivi" ed il file sarebbe risultato incompleto. Quindi occorre sempre ricordarsi di chiudere i file.
Vediamo adesso come fare per recuperare i numeri che abbiamo appena memorizzato sul disco. Anziché inviare delle PRINT, ora dovremo richiedere i dati con delle INPUT:

```
10 OPEN 2, 8, 2, "PROVA, S, R" : REM FILE DI
   LETTURA
20 FOR I = 1 TO 200
30 INPUT # 2, A
40 PRINT A
50 NEXT I
60 CLOSE 2
```

Il funzionamento del programma sarà identico a quello visto prima, tranne che il computer - anziché scrivere - leggerà dal disco.
Un'importanza particolare rivestono, al momento della scrittura dei file, i caratteri separatori tra i vari campi. Per poter leggere i dati con l'istruzione INPUT è infatti necessario che i campi siano registrati, facendo comparire dopo ciascuno di essi un carattere separatore valido, cioè riconosciuto come tale dal disco. I caratteri separatori riconosciuti sono il RETURN (CHR$ (13)) e la virgola. Il primo può essere inserito con la funzione CHR$ (13), oppure facendo terminare la istruzione PRINT senza punteggiatura (come abbiamo fatto noi nell'esempio appena visto).
Il secondo provoca invece delle limitazioni, che causano talvolta perdita di dati o insorgere di errori: poiché richiede una certa esperienza per essere utilizzato, ti conviene non usarlo mai. La cosa migliore, senza entrare troppo in inutili dettagli, è quella di non eseguire mai PRINT od INPUT multiple (cioè del tipo PRINT # 4, A, B, C o INPUT # 4, A, B, C), ma ricorrere sempre a più istruzioni semplici. Per quanto i programmi diventino leggermente più lunghi, la facilità di uso di una simile soluzione permette di restare sempre padroni della situazione, almeno sino a quando non si è acquistata sufficiente familiarità con i segreti dei file.

Un'altra cosa da segnalare è che l'aggiornamento di un file sequenziale può essere ottenuto soltanto riscrivendo completamente il file. Infatti la presenza del carattere EOF, inserito subito dopo l'ultimo record, impedisce l'inserimento di nuovi elementi in fondo alla lista.
Le uniche cose da fare sono allora:
— aprire in lettura il file da aggiornare;
— aprire in scrittura un nuovo file;
— copiare ordinatamente i record dal vecchio file nel nuovo apportando via via le modifiche, le cancellazioni o le aggiunte;
— chiudere alla fine i due file, cancellando il vecchio e cambiando il nome al nuovo file, attribuendogli il nome vecchio.
Vediamo quindi come avremmo dovuto fare per aggiungere altri 100 numeri casuali al file che avevamo creato prima:

```
 10 OPEN 15, 8, 15
 20 OPEN 2, 8, 2, "PROVA, S, R"
 30 OPEN 10, 8, 10, "PROVA2, S, W"
 40 FOR I = 1 TO 200
 50 INPUT # 2, A
 60 PRINT # 10, A
 70 NEXT I
 80 FOR I = 1 TO 100
 90 PRINT # 10, RND (0)
100 NEXT I
110 CLOSE 10 : CLOSE 2
120 PRINT # 15, "SCRATCH : PROVA"
130 PRINT # 15, "RENAME : PROVA = PROVA 2"
140 CLOSE 15
150 END
```

L'esempio che abbiamo appena fatto era abbastanza semplice, visto che sapevamo perfettamente quanti numeri dovevamo trasferire dal vecchio al nuovo file prima di passare alla estrazione di nuovi numeri casuali.
Il più delle volte il numero di letture che possono essere effettuate prima di arrivare all'EOF è invece sconosciuto: tieni inoltre presente che cercare di leggere oltre l'ultimo record provoca un messaggio di errore.
Non è quindi possibile leggere i record "a casaccio".
Un metodo quasi infallibile è allora il seguente. Bisogna modificare il programma, in modo da scrivere nel file, come ultimo record, uno o più caratteri speciali, o facilmente riconoscibili, che tu stabilirai essere il segno di fine-file (per esempio con il nostro file di numeri avremmo potuto scrivere un numero negativo; con un file di stringhe potremmo invece scrivere una stringa di segnalazione con un significato inequivocabile, come "Z * Z * Z").
In questo modo in fase di lettura basterà inserire una istruzione di controllo di fine file, che risulterà verificata al momento in cui il carattere speciale verrà incontrato dal programma.
Ecco un esempio in

proposito. Il programma che segue scriverà sul disco un file, composto da tante stringhe quante ne vorrai introdurre dalla tastiera. Quando deciderai di terminare, basterà che tu batta solo RETURN.
Prima della chiusura del file il computer scriverà automaticamente come ultimo record il nostro carattere speciale "Z * Z * Z".

```
10 OPEN 2, 8, 2,
   "PROVA3, S, W"
20 INPUT "SCRIVI LA
   STRINGA (SOLO * PER
   FINIRE)"; A$
30 IF A$ = "*" THEN
   GOTO 60
40 PRINT # 2, A$
50 GOTO 20
60 PRINT#2,"Z*Z*Z"
70 CLOSE 2
```

Potrai adesso recuperare e rileggere le stringhe che hai appena battuto, con questo secondo programma:

```
10 OPEN 2, 8, 2,
   "PROVA3, S, R"
20 INPUT # 2, A$
30 IF A$="Z*Z*Z"
   THEN CLOSE 2 : END
40 PRINT A$
50 GOTO 20
```

Adesso la conoscenza del numero di elementi appartenenti al file non è più necessaria: il nostro "Z * Z * Z" avverte infatti l'unità centrale di non *tentare di leggere oltre*, perché non esistono altri record. In questo modo ci siamo messi in salvo da un'eventuale possibilità di errore.

Movimento controllato

Con questo programma introduciamo una delle tecniche per controllare un carattere o una animazione per mezzo della tastiera (tasti cursore).
Dalla riga 35 alla riga 60 si effettuano le operazioni di verifica del tasto premuto e si determinano gli spostamenti relativi.

```
10 POKE51,0:POKE52,28:POKE55,0:POKE56,28
15 DATA60,126,219,255,195,126,90,195
20 FORC=7168TO7679:POKEC,0:NEXTC
25 FORC=7168TO7175:READA:POKEC,A:NEXTC
30 LETX=0:LETY=0:PRINT" ♥ @ ":POKE36869,255
35 GETT$:IFT$="   "THENGOTO35
40 LETT=ASC (T$):LECT=32:GOSUB75
45 IFT=17ANDY<22THENLETY=Y+1
50 IFT=29ANDX<21THENLETX=X+1
55 IFT=145ANDY>0THENY=Y—1
60 IFT=157ANDX>0THENLETX=X—1
65 IFT=83THENPOKE36869,240:END
70 LETC=0:GOSUB75
75 LETA=X+Y*22:POKE7680*A,C
80 POKE38400+A,6:RETURN
```

VIDEOESERCIZI

Dopo aver esaminato attentamente il listato rispondi alle domande verificando le tue risposte personalmente alla tastiera.

```
 10 GOSUB 50
 20 GOSUB 10
 30 RETURN
 40 END
 50 S=32400:S=S+1
 60 PRINT "QUANTE VARIABILI..."
 70 GOTO 30
 80 IF S<56 THEN RETURN
 90 RETURN
100 K=K+1:GOSUB 100:REM PER CONTARE
    QUANTI GOSUB OCCORRONO PER RIEMPIRE LO STACK
```

Quale errore causa questo programma?

La "IF" in linea 80 potrà mai essere vera?

Il programma si ferma con la "END" ... o no?

Quanti GOSUB può sopportare il "GOSUB STACK"?